<u>Chatbot</u>

Agenda

• Overview

“[AI] is going to change the world more than anything in the history of mankind. More than electricity.”— AI oracle and venture capitalist Dr. Kai-Fu Lee, 2018

In today's mobile world everyone of us have been users of the technology AIML. From the video predictions, shopping, service centres, social media, surveillance, food delivery, transportation, self driving and so on. Interestingly we are as well the producers of these data.

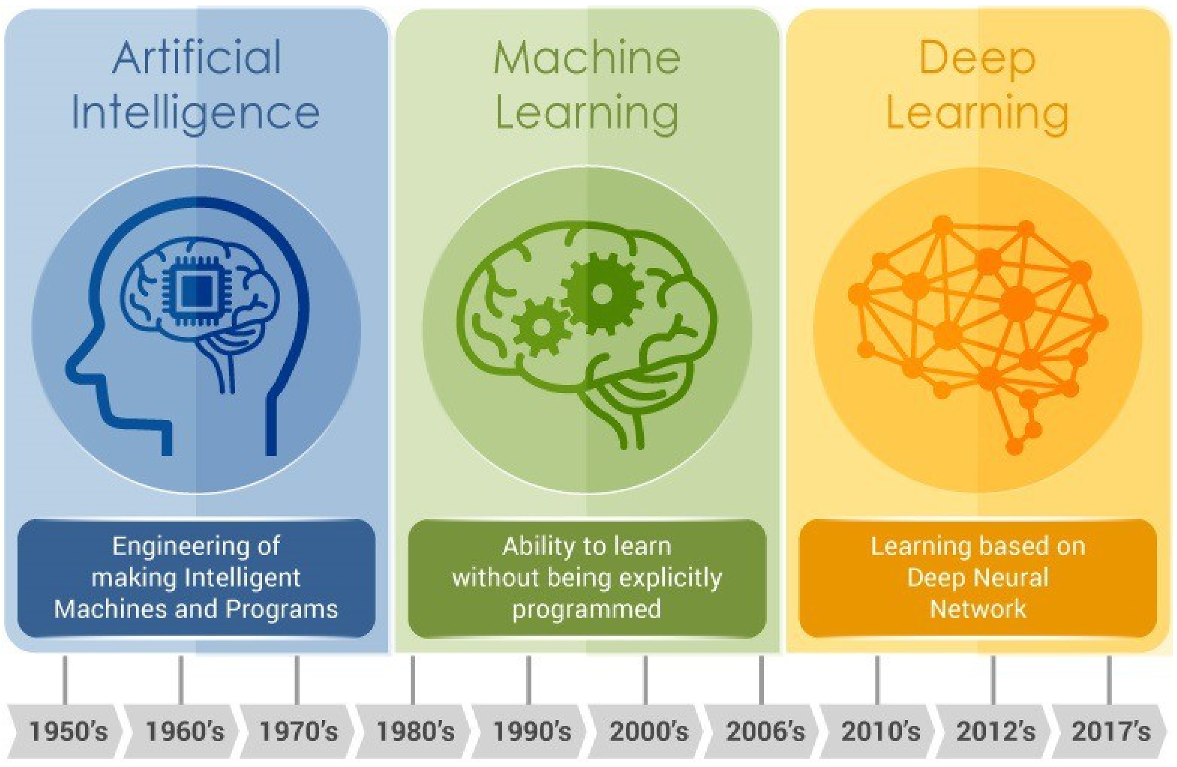
AI's impact is everywhere, below are TOP 10 industries which will have revolutionise.

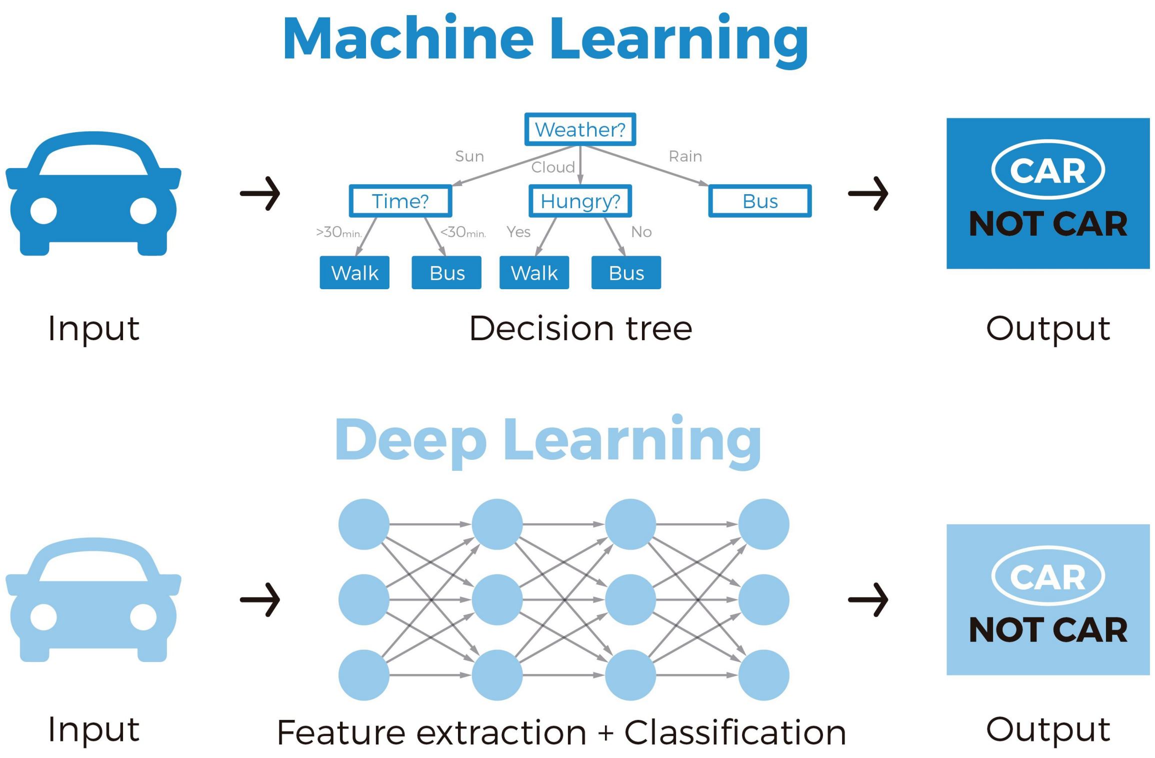
- Transportation
- Manufacturing
- Healthcare
- Education
- Media
- Customer service
- Financial
- E-Commerce
- Sports
- Agriculture

We would briefly see today what is AIML and its applications. Understand NLP, as it is one of disciplines. NLP helps in understanding, interpret and manipulate human language. Build a chatbot from scratch using Rasa framework.

- **Artificial intelligence**

Artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind.


Artificial Intelligence
Engineering of making Intelligent Machines and Programs
Machine Learning
Ability to learn without being explicitly programmed
Deep Learning
Learning based on Deep Neural Network
1950's
1960's
1970's
1980's
1990's
2000's
2006's
2010's
2012's
2017's



- **Machine Learning**

"[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed." Arthur Samuel 1959.

What is machine learning?

- Is a process of enabling a computer based system to learn to do tasks based on well defined statistical and mathematical methods
- The ability to do the tasks come from the underlying model which is the result of the learning process. Sometimes the ability comes from an mathematical algorithm
- The model is generated from huge volume of data, huge both in breadth and depth reflecting the real world in which the processes are performed
- The more representative data is of the real world, the better the model would be. The challenge is how to make it a true representative

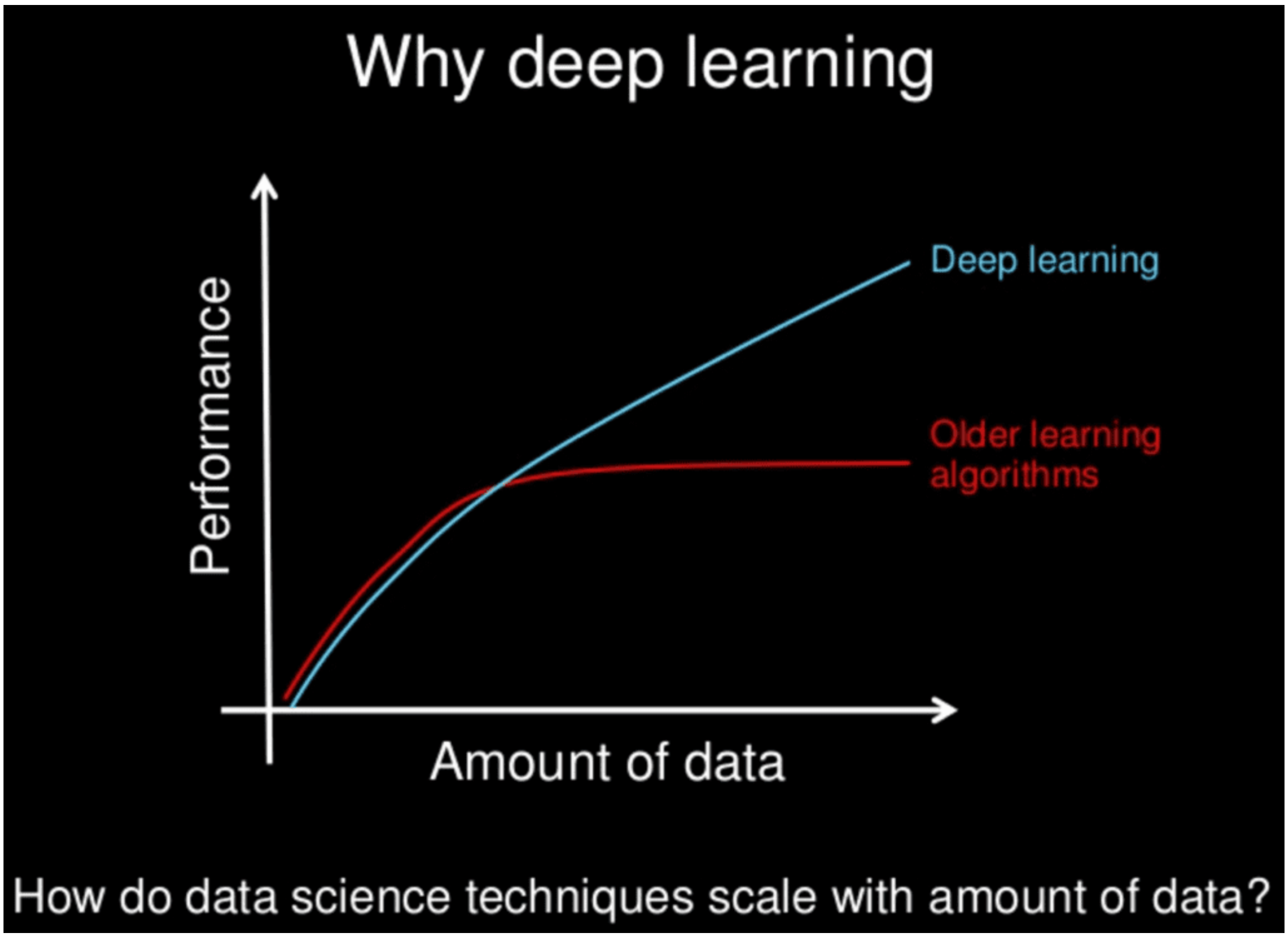
What do machine learning algorithms do?

- Search through data to look for patterns
- Patterns in form of trends, cycles, associations, classes etc.
- Express these patterns as mathematical structures such as probability equations or polynomial equations

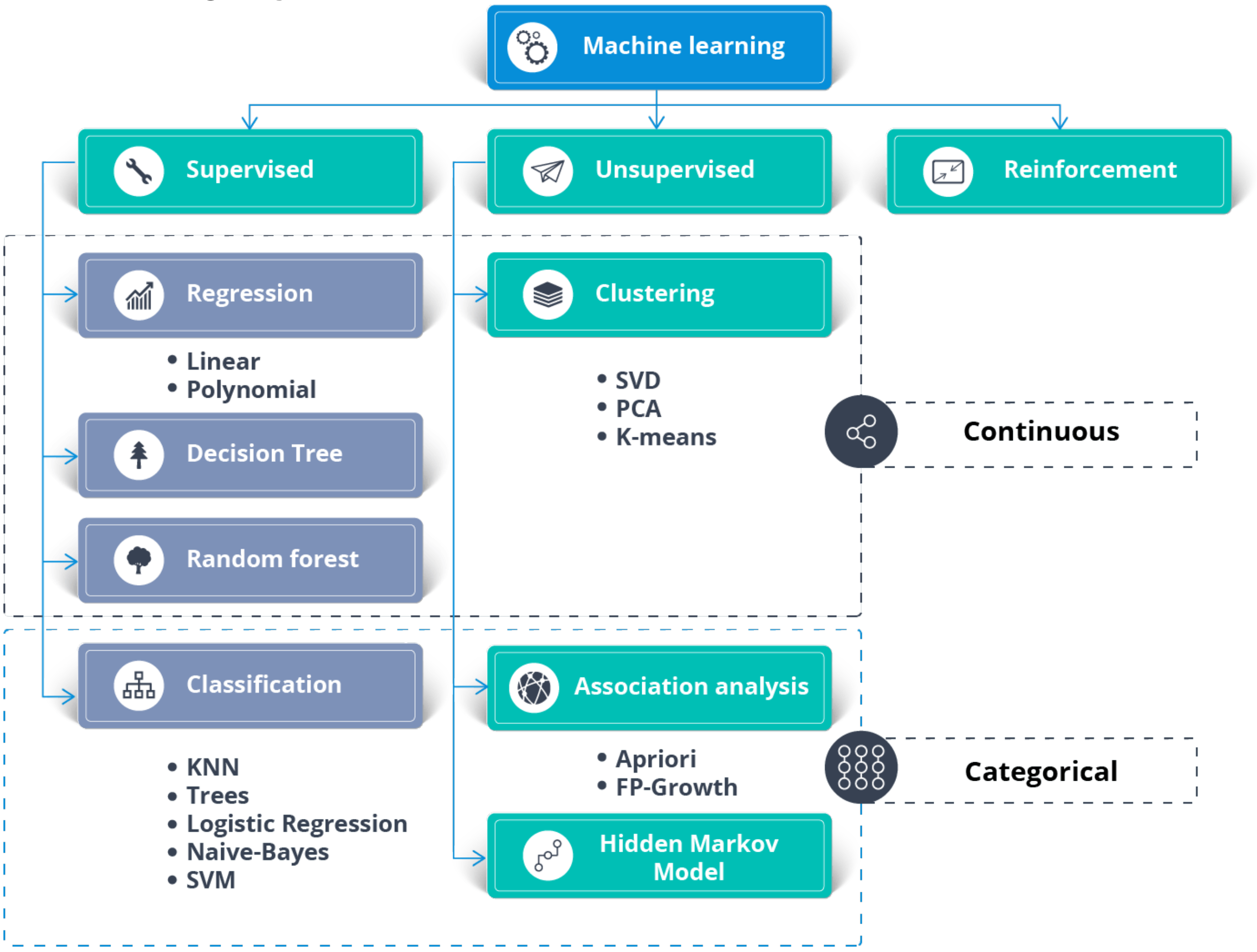
Where are machine learning based systems used (examples only)

- Fraud detection
- Sentiment analysis
- Credit risk management
- Prediction of equipment failures
- New pricing models / strategies
- Network intrusion detection
- Pattern and image recognition
- Email spam filtering

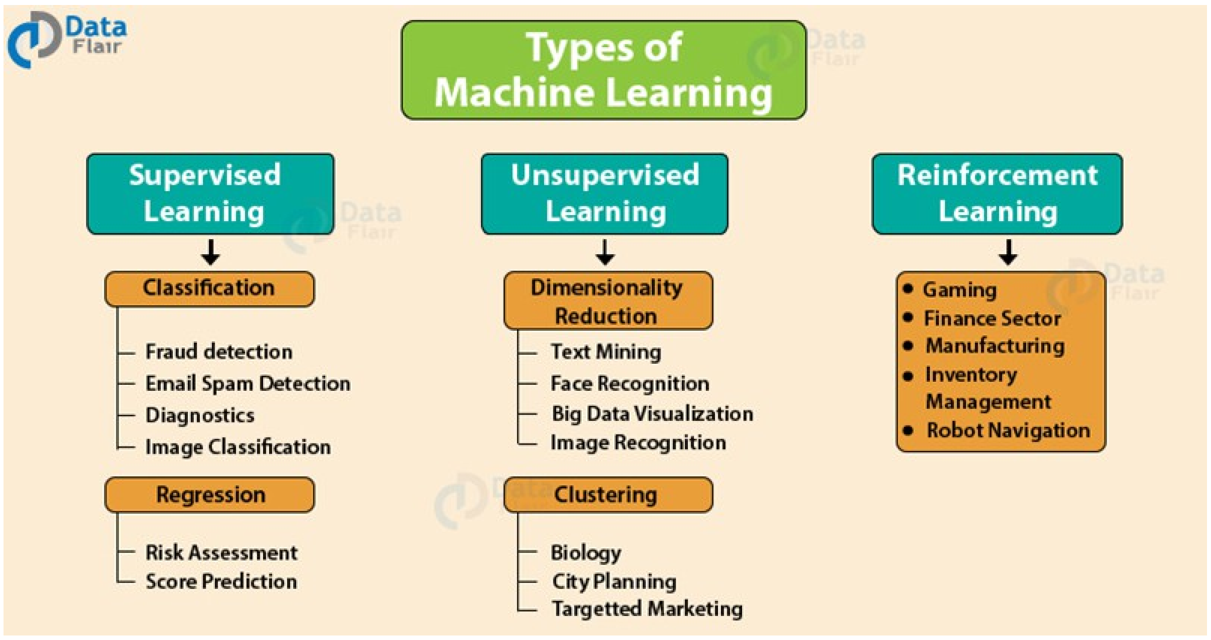
- **Deep Learning**
  - Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example.
  - Deep learning is a key technology behind driverless cars, enabling them to recognise a stop sign, or to distinguish a pedestrian from a lamppost.
  - It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It’s achieving results that were not possible before.
  - In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.
  - Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video.
  - Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.



• **Category**



Machine learning
Supervised
Unsupervised
Reinforcement
Regression
• Linear
• Polynomial
Decision Tree
Random forest
Clustering
• SVD
• PCA
• K-means
Continuous
Classification
• KNN
• Trees
• Logistic Regression
• Naive-Bayes
• SVM
Association analysis
• Apriori
• FP-Growth
Hidden Markov Model
Categorical



Data Flair
Types of Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Classification
Fraud detection
Email Spam Detection
Diagnostics
Image Classification
Regression
Risk Assessment
Score Prediction
Dimensionality Reduction
Text Mining
Face Recognition
Big Data Visualization
Image Recognition
Clustering
Biology
City Planning
Targetted Marketing
• Gaming
• Finance Sector
• Manufacturing
• Inventory Management
• Robot Navigation

- **Computer vision**
    - Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.
    - Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.
    - Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.
    - Computer vision is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow. It is expected to reach USD 48.6 billion by 2022
    - Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.
    - Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).
    - Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will “look” at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

- A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is “seeing.” The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

- **Natural language processing**
  - Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.
  - Today’s machines can analyse more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data that’s generated every day, from medical records to social media, automation will be critical to fully analyse text and speech data efficiently.
  - Sample Projects
    - Sentiment Analysis
      - customer reviews
      - customer segmentation
      - anomaly detection
      - product improvement
    - Topic Modelling
      - coming up with new topics from the text
      - using those topics to assign new supervised learning labels
      - insights that are too difficult to find from manual searching
    - Text Categorisation
      - categorising animal specials
      - categorising fake news
      - categorising bank transactions

- **Chatbot**

A chatbot is a computer program that simulates human conversation through voice commands or text chats or both. Chatbot, short for chatterbot, is an artificial intelligence (AI) feature that can be embedded and used through any major messaging applications.

Chatbots, also called chatterbots, is a form of artificial intelligence (AI) used in messaging apps.

This tool helps add convenience for customers—they are automated programs that interact with customers like a human would and cost little to nothing to engage with.

Key examples are chatbots used by businesses in Facebook messenger, or as virtual assistants, such as Amazon's Alexa.

Chatbots tend to operate in one of two ways—either via machine learning or with set guidelines.

- **Rasa**

Rasa Open Source supplies the building blocks for creating virtual assistants. Use Rasa to automate human-to-computer interactions anywhere from websites to social media platforms.

Rasa supplies conversational AI infrastructure for a global community of developers, providing the tools to build chat- and voice-based contextual assistants.

Rasa is powered by open source software and runs in production everywhere from startups to Fortune 500s, across industries like healthcare, financial services, retail, and insurance.

Rasa Open Source provides three main functions. Together, they provide everything you need to build a virtual assistant:

- **Natural Language Understanding**
  Convert raw text from user messages into structured data. Parse the user’s intent and extract important key details.
- **Dialogue Management**
  Machine learning-powered dialogue management decides what the assistant should do next, based on the user’s message and context from the conversation.
- **Integrations**
  Built-in integration points for over 10 messaging channels, plus endpoints to connect with databases, APIs, and other data sources.

- Installation
  The first step before we proceed, lets install rasa.

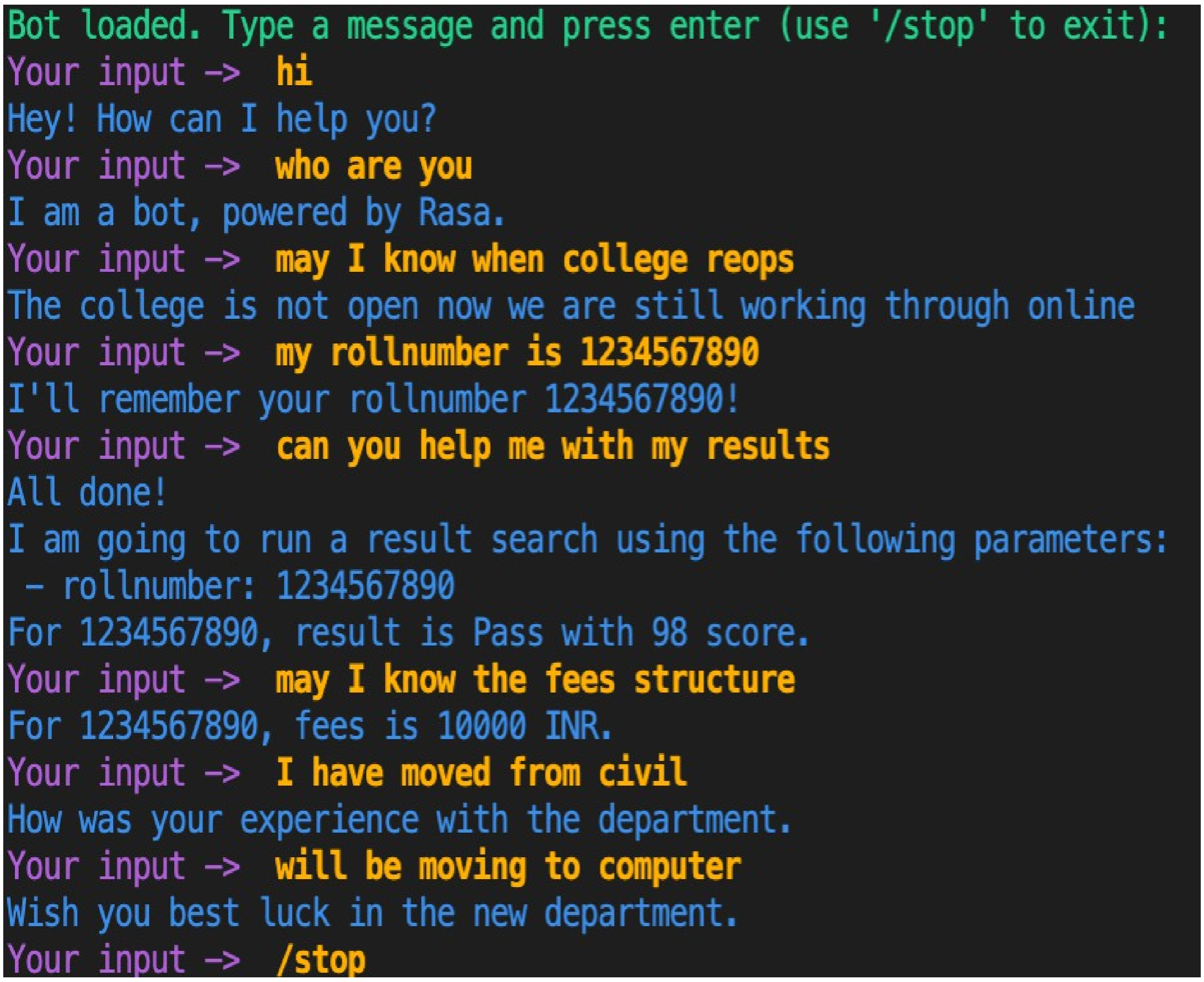
```
% python -m venv env
% source activate
% pip install rasa OR pip install rasa-x -i https://pypi.rasa.com/simple
% rasa init
% update config.yml and endpoint.yml [optional]
% rasa train
    % rasa train nlu
    % rasa train core
% rasa shell
% rasa run actions
% rasa data validate
```

- NLU
  - Intent
  - Training data
  - Lookup
  - Regular Expression
  - Entities
  - Synonyms
  - Entities Roles and Groups

- Core
  - Stories
  - Checkpoints
  - Rules
  - Action
  - Events
    - Slots
    - Form
  - Rules
  - Domain
  - Config

- Rasa X

Sample Conversation:



```
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  hi
Hey! How can I help you?
Your input ->  who are you
I am a bot, powered by Rasa.
Your input ->  may I know when college reops
The college is not open now we are still working through online
Your input ->  my rollnumber is 1234567890
I'll remember your rollnumber 1234567890!
Your input ->  can you help me with my results
All done!
I am going to run a result search using the following parameters:
 - rollnumber: 1234567890
For 1234567890, result is Pass with 98 score.
Your input ->  may I know the fees structure
For 1234567890, fees is 10000 INR.
Your input ->  I have moved from civil
How was your experience with the department.
Your input ->  will be moving to computer
Wish you best luck in the new department.
Your input ->  /stop
```

Scenario:

- College timings

nlu.yml

```
- intent: timings
examples: |
- I would like to college timings
- college time please
- what would the class start and end time
- when college reopen
```

stories.yml

```
- story: college time path
steps:
- intent: timings
- action: utter_timings
```

domain.yml

```
intents:
- timings

responses:
utter_timings:
- text: "The college is not open now we are still working
through online"
```

- Course duration for department

nlu.yml

```
- lookup: department
examples: |
- civil
- mechanical
- computer
- textile
- printing

- intent: course_duration
examples: |
- what is course duration for [civil]{"entity": "department"}
- would like to know the [computer]{"entity": "department"}
course tenure
```

stories.yml

```
- story: college course duration
steps:
- intent: course_duration
- action: action_course_duration
```

domain.yml

```
intents:
- course_duration

actions:
- action_course_duration

entities:
- department

responses:
utter_course_duration:
- text: "For {department} course is of {duration} months"
```

actions.py

```
class ActionCourseDuration(Action):

def name(self) -> Text:
     return "action_course_duration"

def run(self, dispatcher: CollectingDispatcher,
tracker: Tracker,
domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

     # It will return array of entities
     entities = tracker.latest_message['entities']
     print(entities)

     course_duration = {
     'civil': 10,
     'computer': 12,
     'mechanical': 14,
     'printing': 16,
     'textile': 18
     }
```

```
entity_department = None

# Iterating through the array to retrieve the desired
entity
for e in entities:
     if e['entity'] == "department":
          entity_department =
          str(e['value']).lower().strip()
          duration =
          course_duration.get(entity_department, 0)

dispatcher.utter_message(
response="utter_course_duration",
department=entity_department,
duration=duration
)
#dispatcher.utter_message(text="Hello World!")

return []
```

- Exam Results

nlu.yml

```
- regex: rollnumber
examples: |
- \d{10,30}

- intent: get_roll_number
examples: |
- my roll number is [1234567891](rollnumber)
- This is my roll number [1234567891](rollnumber)
- [1234567891](rollnumber)

- intent: request_result
examples: |
- may I know the exam results
- can you please help me to know if I have passed
- am I all clear
```

rules.yml

```
- rule: activate result form
steps:
- intent: request_result # intent that triggers form
activation
- action: result_form # run the form
- active_loop: result_form # this form is active

- rule: submit form
condition:
- active_loop: result_form # this form must be active
steps:
- action: result_form # run the form
- active_loop: null # the form is no longer active because it
has been filled
- action: utter_submit # action to take after the form is
complete
- action: utter_slots_values # action to take after the form
is complete
- action: action_show_result
```

domain.yml

```
intents:
- get_roll_number
- request_result

actions:
- action_show_result

forms:
     result_form:
          required_slots:
               rollnumber:
               - type: from_entity
                 entity: rollnumber

slots:
     rollnumber:
          type: any
          auto_fill: false
          influence_conversation: false

entities:
- rollnumber

responses:
utter_result:
- text: "For {roll}, result is {result} with {score} score"

utter_ask_rollnumber:
- text: "Please provide your roll number"

utter_submit:
- text: "All done!"

utter_slots_values:
- text: "I am going to run a result search using the
following parameters:\n
- rollnumber: {rollnumber}"
```

actions.py

```
class ActionShowResult(Action):

def name(self) -> Text:
return "action_show_result"

def run(self,
dispatcher: CollectingDispatcher,
tracker: Tracker,
domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

     roll = tracker.get_slot("rollnumber")
     print("Rollno: ", roll)
     if( roll ):
          score = 98
          roll = 100
     else:
          score = -1
          roll = 0

     result = "Fail"
     if score >= 50:
          result = "Pass"

     dispatcher.utter_message(
          response="utter_result",
          score=score,
          roll=roll,
          result=result
     )

     return []
```

- Fees Enquiry
  - Provide if only roll number was provided

nlu.yml

```
- regex: rollnumber
examples: |
- \d{10,30}

- intent: get_roll_number
examples: |
- my roll number is [1234567891](rollnumber)
- This is my roll number [1234567891](rollnumber)
- [1234567891](rollnumber)

- intent: fees_enquiry
examples: |
- may I know the fees structure
- how much fees do I need to pay
- do I have any pending fees to be paid
```

stories.yml

```
- story: save rollnumber
steps:
- intent: get_roll_number
- action: action_save_roll_number
```

rules.yml

```
- rule: Only say `fees` if the user provided a rollnumber
condition:
- slot_was_set:
- rollnumber: true
steps:
- intent: fees_enquiry
- action: action_fees_details
```

domain.yml

```
intents:
- fees_enquiry

actions:
- action_fees_details

entities:
- department
- rollnumber

slots:
     rollnumber:
          type: any
          auto_fill: false
          influence_conversation: false

responses:
utter_fees:
- text: "For {roll}, fees is {fees} INR."
```

actions.py

```
class ActionShowFeesStructure(Action):

     def name(self) -> Text:
          return "action_fees_details"

     def run(self,
     dispatcher: CollectingDispatcher,
     tracker: Tracker,
     domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

          roll = tracker.get_slot("rollnumber")
          print("Rollno: ", roll)
          fees = 0
          if( roll ):
          fees = 10000

          dispatcher.utter_message(
          response="utter_fees",
          fees=fees,
          roll=roll
          )

          return []


class ActionReceiveRollNumber(Action):

     def name(self) -> Text:
          return "action_save_roll_number"

     def run(self, dispatcher: CollectingDispatcher,
     tracker: Tracker,
     domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

          #text = tracker.latest_message['text']
          entities = tracker.latest_message['entities']

          roll = None
          for e in entities:
          if e['entity'] == "rollnumber":
          roll = str(e['value']).lower().strip()

          dispatcher.utter_message(text=f"I'll remember your
          rollnumber {roll}!")
          return [SlotSet("rollnumber", roll)]
```

- Change of department request

nlu.yml

```
- intent: department_have_been_changed
examples: |
- I have changed from [civil]{"entity": "department", "role":
"from"}
- Have moved from [civil]{"entity": "department", "role":
"from"}

- intent: department_going_to_change
examples: |
- I am going to [civil]{"entity": "department", "role": "to"}
department
- I am changing to [civil]{"entity": "department", "role":
"to"} department
- Will be moving to [civil]{"entity": "department", "role":
"to"} course
```

stories.yml

```
- story: The student moving from another department
steps:
- intent: department_have_been_changed
entities:
- department: Civil
role: from
- action: utter_ask_about_experience

- story: The student is going to another department
steps:
- intent: department_going_to_change
entities:
- department: Computer
role: to
- action: utter_wish_luck
```

domain.yml

```
intents:
- department_have_been_changed
- department_going_to_change

responses:
utter_ask_about_experience:
- text: "How was your experience with the department."

utter_wish_luck:
- text: "Wish you best luck in the new department."
```

Best Practices

- Real world test data
- Test conversation
- Managing conversation data files

References:

https://rasa.com/docs/
https://github.com/RasaHQ/rasa-form-examples